

**RL-TR-95-214**  
**Final Technical Report**  
**October 1995**



# **IIPF EXPERT SYSTEM EXPERIMENTS (IIPLESE)**

**Planning Research Corporation**

**Steve Barth and Mike Scully**

*APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.*

19960419 158

**Rome Laboratory**  
**Air Force Materiel Command**  
**Rome, New York**

**DTIC QUALITY INSPECTED 1**

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be releasable to the general public, including foreign nations.

RL-TR-95- 214 has been reviewed and is approved for publication.

APPROVED:



JOHN M. PIROG  
Project Engineer

FOR THE COMMANDER:



RONALD W. CWIRKO  
Acting Director  
Intelligence & Reconnaissance Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify Rome Laboratory/ ( IRDS ), Rome NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE October 1995		3. REPORT TYPE AND DATES COVERED Final Aug 91 - Sep 94	
4. TITLE AND SUBTITLE  IIPF EXPERT SYSTEM EXPERIMENTS (IIPLESE)				5. FUNDING NUMBERS C - F30602 91-C-0109 PE - 62702F PR - 4594 TA - 16 WU - 40	
6. AUTHOR(S)  Steve Barth and Mike Scully					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Planning Research Corporation 1500 PRC Drive McLean VA 22102				8. PERFORMING ORGANIZATION REPORT NUMBER  N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Rome Laboratory/IRDS 32 Hangar Rd Rome NY 13441-4114				10. SPONSORING/MONITORING AGENCY REPORT NUMBER  RL-TR-95-214	
11. SUPPLEMENTARY NOTES  Rome Laboratory Project Engineer: John M. Pirog/IRDS/(315) 330-3222					
12a. DISTRIBUTION/AVAILABILITY STATEMENT  Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Under the Intelligence Information Processing Laboratory Expert Systems Experiments (IIPLESE) effort, tests and experiments were conducted on applications that are part of the Intelligence Predictive Assessment System (IPAS) 2000 concept. The IPAS 2000 concept was developed to serve as a plan for integration of R&D applications with IDHS database management systems. The tests and experiments performed under IIPLESE were the initial steps toward this integration. The IIPLESE effort had the important purpose of testing and experimenting with the Program 6 applications to determine how they can be used efficiently to share data and knowledge within a common framework that follows the principals of the IPAS 2000 concept.					
14. SUBJECT TERMS Testing, Inegration, Message processing, Speech recognition, Configuration management				15. NUMBER OF PAGES 40	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL		

# Contents

---

<b>Preface .....</b>	<b>ii</b>
<b>1. Introduction .....</b>	<b>1-1</b>
<b>2. Task Results Summary .....</b>	<b>2-1</b>
2.1 Configuration Management .....	2-1
2.2 The IPAS 2000 "Vertical Slice" .....	2-2
2.3 Message Filtering with WIDE .....	2-7
2.4 GIP Tests and Enhancements .....	2-9
2.5 Speech Understanding Integration .....	2-10
2.5.1 Speech Recognition Systems .....	2-10
2.5.2 Speech Recognition Capabilities and Limitations .....	2-12
2.5.3 Speech Recognition Products and Environments .....	2-13
2.5.3.1 IN3 Voice Command .....	2-14
2.5.3.2 HM2007 Speech Recognition Chip .....	2-14
2.5.3.3 Entropic HTK Toolkit .....	2-14
2.5.3.4 DragonVoice Tools .....	2-14
2.5.3.5 BBN's HARK™/Delphi Systems, and the SPLINT Domain .....	2-15
2.5.4 Integrating Speech Recognition Into IPAS 2000 Applications .....	2-16
2.6 ART/AMUS Integration and Testing .....	2-20
<b>3. Additional Accomplishments .....</b>	<b>3-1</b>
<b>4. What's Next .....</b>	<b>4-1</b>

# Preface

---

This document is the final report for the Intelligence Information Processing Laboratory Expert Systems Experiments effort, IIPLESE, contract number F30602-91-C-0109. The IIPLESE effort was aimed at testing and integrating a number of prototype systems developed for Rome Laboratory, Intelligence and Reconnaissance Division Technology Branch, RL/IRDS. Mr. John Pirog, RL/IRDS, was the RL project engineer.

The prime contractor for the IIPLESE effort was PRC Inc., with subcontractors of Sterling Software, Grumman Data Systems, and MRJ Inc. Mr. Chris A. Boehm was the PRC project manager. Most of the IIPLESE effort was performed on-site at RL by Mr. Stephen W. Barth and Mr. Michael C. Scully of PRC, Inc., and Mr. Jeff Penatzer of Sterling Software, under the technical direction of Mr. Barth. Additional effort was performed by Dr. Stefan Shrier of MRJ, Inc., and formerly of Grumman Data Systems. The work was performed from August, 1991, through September, 1994.

Results from the IIPLESE work are documented in several technical reports, and this final report. The principal accomplishments were: the testing and improvement of the Warning Information Dissemination Experiment (WIDE) software for message filtering, the thorough testing and integration of the Generic Intelligence Processor (GIP) software for message processing, and the experimental implementation of a portion of the IPAS 2000 conceptual architecture that performs fully automated filtering, processing, data extraction, and presentation of results for analysis of message data.

# 1. Introduction

---

Under the Intelligence Information Processing Laboratory Expert Systems Experiments effort, IIPLESE, tests and experiments were conducted on applications that are part of the Intelligence Predictive Assessment System (IPAS) 2000 concept [1]. The IPAS 2000 concept was developed by the Rome Laboratory Intelligence and Reconnaissance Directorate Technology Branch, RL/IRDS, to serve as a plan for integration of Program 6 applications with IDHS database management systems. The tests and experiments performed under IIPLESE were the initial steps toward this integration. Program 6 applications were analyzed for robustness and inter-operability, and integrated within a common framework for sharing data and results.

The IPAS 2000 concept divides applications into two categories: information resource management and analytical tools. Information resource management applications filter and extract data from text messages. Analytical tool applications assist the analyst in understanding and visualizing data. The IDHS databases serve as the repository of the collected data and analytical results. Under the IPAS 2000 concept, both sets of applications are to be integrated within a "plug and play" framework. This means that applications that perform the same function and meet the same interface specifications should be interchangeable within the framework.

The IIPLESE effort had the important purpose of testing and experimenting with the Program 6 applications to determine how they can be used efficiently to share data and knowledge within a common framework that follows the principals of the IPAS 2000 concept. The key Program 6 applications used for the IIPLESE effort were

<i>ART/AMUS</i>	the Advanced Reasoning Tool, Advanced Reasoning Tool Message Understanding System for data extraction [2]
<i>GIP</i>	the Generic Intelligence Processor for building message processing applications and integrating natural language understanding (NLU) techniques [3]
<i>TAS</i>	the Temporal Analysis System for presenting and analyzing data for situation assessment [4]
<i>WIDE</i>	the Warning Information Dissemination Experiment for message filtering [5]

In addition, other important off-the-shelf software packages or systems used for the IIPLESE experiments were:

<i>eM</i>	the PRC Event Manager for distributed processing using a publish and subscribe framework[6, 7]
<i>PLUM</i>	BBN's Probabilistic Language Understanding Module for natural language understanding (NLU) [8]
<i>DAWS/ DMFE</i>	the Defense Automated Warning System, and Defense Automated Warning System Message Front End database management system for I&W data [9]

The three main tasks accomplished under IIPLESE were integration of WIDE, GIP, PLUM, and TAS to form a "vertical slice" of the IPAS 2000 concept, the thorough testing of GIP, and the thorough testing, evaluation, and improvements to WIDE. Each of these tasks is described in a separate technical report and a published paper [10, 11, 12, and 13]. Part of the IIPLESE effort was spent in configuration management of the workstations and LANs in the Intelligence Information Processing Facility (IIPF) and Intelligence Information Processing Support Facility (ISF) to support these and other tests and experiments. That work is documented in two technical reports [14, 15]. Other tasks that were performed are also described in this report.

Section 2 of this report describes the major technical tasks performed under IIPLESE. Section 3 describes additional accomplishments. Section 4 discusses ideas for additional work that were prompted by the IIPLESE tasks, and Section 5 lists references.

## **2. Task Results Summary**

---

The three main tasks accomplished under IIPLESE were integration of WIDE, GIP, PLUM, and TAS to form a "vertical slice" of the IPAS 2000 concept, the thorough testing of GIP, and the thorough testing, evaluation, and improvements to WIDE. The basis for these tasks was the establishment of well-managed local area workstation networks in the Intelligence Information Processing Facility (IIPF), and the Intelligence Information Processing Facility Support Facility (ISF). In addition, other tasks examined the integration of speech understanding capabilities, and the integration of ART/AMUS for natural language understanding.

### **2.1 Configuration Management**

During the course of the IIPLESE effort, the ISF evolved rapidly from a few standalone PCs and Sun 3 workstations to a well-administered LAN consisting of over a dozen workstations. The Program 6 area in Vault 918 of the IIPF was also reorganized and several new workstations were installed. As a result of IIPLESE configuration management activities in these areas, a testbed environment to support classified and unclassified development and testing for IRD Technology Branch was created. During the IIPLESE effort, this environment provided the basis for porting applications to the IIPF for use with classified data, and many other tasks and experiments. The ISF and IIPF Program 6 facilities are described in detail in two technical reports [14, 15]. It is important to note that the term "configuration management", under IIPLESE, meant performing not only system administration tasks, but also software development when and where it was needed. Some of the main features and capabilities that were added during the IIPLESE effort are:

- InterNet access to the ISF
- Use of NIS to ease system administration chores
- Installation of applications and utilities to provide an in-house development environment.
- Installation and improvements to contract applications and demos.

During the IIPLESE effort, the ISF and IIPF Program 6 areas also supported a broad range of IRD activities - Summer projects by students and visiting professors, development of PSIDS, development and integration of applications under the Work Station Support Segment (WSS), development of the Air Force Client-Server Environment (CSE), the Temporal Analysis System Program (TAS), integration of applications with the Defense Automated Warning System (DAWS) and the DAWS Message Front End (DMFE). The testbed environment developed in the ISF and IIPF Program 6 area was used to support all of these activities, in addition to the experiments and tasks conducted under IIPLESE.

## **2.2 The IPAS 2000 "Vertical Slice"**

The IPAS 2000 "Vertical Slice" demonstration (VS Demo) provided a prototype implementation of the IPAS 2000 conceptual architecture. WIDE, GIP, and PLUM were used as Information Resource Management Tools, and two versions of the TAS application, C2TAS and AirTAS, were used as Analytical Tools. The IDHS databases were simulated with a single small database. The VS Demo was a important first attempt at integrating these applications within the IPAS 2000 framework.

The VS Demo was developed with several goals in mind. Primarily, it was intended as an experiment in system integration using the publish and subscribe mechanisms provided by the PRC Event Manager (eM), and as a development framework for other IIPLESE tests and experiments. It was also intended as an experiment to examine the problems and issues of integrating the applications and systems that are part of the IPAS 2000 concept. Finally, it was intended to examine whether a current state-of-the-art NLU system (BBN's PLUM) could be used to supply data extracted from free text messages to a fielded analytical tool (GTE's TAS).

The objective for the VS Demo was to show free-text messages being automatically processed from raw input to end-user application data records, without user intervention. The system simulates the delivery of input message

data distributed over time, as it might arrive in an operational setting, from a wire-service feed, from the Communications Support Processor (CSP), or as the result of a query to the Modular Architecture for Exchange of Intelligence (MAXI). Messages are then filtered using the statistically-based concept-matching techniques from the Warning Information Dissemination Experiment, (WIDE). Messages that do not pass through the filters are archived for possible later review. Those messages selected for further processing are sent to a Message Processing Application (MPA) developed using the Generic Intelligence Processor (GIP). The MPA extracts the text portion of the message and sends it to a Natural Language Understanding (NLU) system, BBN's Probabilistic Language Understanding Module (PLUM). The MPA then formats and translates the data returned by the NLU to provide records for insertion into a central database. The central database, which has been named the VS Database, plays the role of the IPAS 2000 databases that serve as a boundary between tools and applications for message processing and those for analysis. Insertion of new data into the database triggers forwarding of data elements to two end-user applications for situation assessment, the Command and Control Timeline Analysis System (C2TAS) and the Air Timeline Analysis System (AirTAS). C2TAS and AirTAS are versions of GTE's Timeline Analysis System (TAS) oriented, respectively, toward command and control events and events involving movements of aircraft. Each version of TAS has its own database representing the types of events to be analyzed. TAS users manipulate the data for analyses to support various kinds of situation assessment in such areas as counter-narcotics, anti-terrorism, and Indications and Warnings.

The VS Demo system is made up of the applications described above plus various interface processes. It is implemented on five UNIX workstations in the Intelligence Information Processing Facility Support Facility (ISF): two SPARCstation 10s, a SPARCstation 2, a SPARCstation 1+, and a DECstation 5000/200.

The VS Demo system has been tested with sample message data selected from the wire service reports used at the Third Message Understanding Conference (MUC-3), and twenty SPOT messages. This data was selected precisely because the available version of PLUM had been

developed for BBN's participation in MUC-3 and for a demonstration dealing with SPOT messages. It was not expected that PLUM would be able to process the data without errors or ambiguity, but the idea was to obtain insight into whether at least some information useful to an analytical tool could be automatically extracted from free text.

The publish and subscribe paradigm, where client processes request certain types of named information from the processing environment (subscribe), and produce named information that is distributed to other client subscribers (publish), proved to be a very flexible and easy way to integrate the various systems in the VS Demo. Two main advantages of this approach are that applications publishing or subscribing to information can be hosted anywhere on the network, and that multiple applications can subscribe to or publish the same types of information. Since the topology of workstations running as eM servers provides connectivity to any other local area network (LAN) host that is an eM server, no host specific information is needed in integrating the VS Demo applications. The only information required is the naming conventions for information that is to be distributed. Since multiple subscribers can obtain the same types of information, it is easy to arrange for parallel distributed processing of applications to improve throughput or provide alternative results.

Because of the publish and subscribe paradigm, the VS Demo provides a convenient framework for further tests and application integration within the IPAS 2000 conceptual architecture. The value of the VS framework is that it enables an examination of how the various components can best fit together at the information-flow level. For example, it is clear from the current arrangement of applications that some amounts of redundant processing occur. Both the message filtering processes and the GIP MPA parse the header information of the message, and both the message filtering processes and the NLU applications perform part-of-speech tagging on the text of the message. Throughput and performance issues can also be examined to identify processing bottlenecks or where more efficient use of on-line storage is required. As expected, the bottleneck in processing for the current system is in the NLU component; however, some improvement may be realized by taking

advantage of the publish and subscribe paradigm's inherent capability for parallelism. The current system takes the simple approach of using multiple NLU systems to handle different types of messages in parallel.

Since NLU performance is far from perfect, a significant deficiency noted in the current system is the lack of some means of determining the confidence of information extracted from the text. PLUM uses confidence factors, but does not currently export them in the output processed by GIP. Thus in cases where PLUM offers alternative interpretations of text, represented by multiple templates of possibly conflicting information, there is no means for the system to select the "best" interpretation, or to provide the analytical applications with a ranked list of interpretations. This problem becomes more apparent when data from the VS Demo is displayed using the current analytical applications, C2TAS and AirTAS. Since these applications perform no automatic data fusion or data base consistency checking, the user must sort out what may be sets of conflicting data records.

Two other integration issues arose during development of the VS Demo: the need for data translation, and the lack of interfaces for automated processing. The data translation or consistent interpretation problem is well-known in any system where data moves between one data base and another. Ultimately, the IPAS 2000 concept seeks to address this problem by making the central databases; CATIS, DAWS, MAXI, and XIDB, the standard for both message processing and analytical applications; however, because of unique requirements for processing, most analytical applications will still require their own representations of data (e.g., TAS requires a data representation suited for efficient timeline manipulation via it's graphic user interface). The current VS Demo puts the burden of translation on the information subscriber; that is, a process receiving published information is responsible for translating it into any local representation. This approach could eventually take advantage of the Knowledge Exchange Language (KXL) [16] developed for the Cooperating Knowledge-Based Architecture (CKBA) effort, which provides a common language and translation mechanisms for publishers and subscribers of information. Another approach would be to use the translation module from GIP, which could be extracted and used as a front-end for each application. The GIP

translation module can be easily configured to handle application-specific data format requirements.

None of the applications used for the VS Demo were designed for automated input using publish and subscribe techniques. Where applications acquire input from files and provide output as files, as is the case for GIP and WIDE, simple eM processes were written to provide "wrappers" around the application. Some applications, like TAS, however, are essentially designed to handle input only through the user interface. This presents a problem because essential features needed for the data may be embedded in the code or in the design of the UI. For example, the TAS user, as part of the process of entering event data, normally assigns an icon to represent an event by choosing one from a palette. Another, similar type of problem occurs when the UI checks for valid fields. Event data automatically entered may bypass the validity checks and cause problems if the user later tries to edit it. The current VS Demo really only bypasses handling these kinds of problems. Choices of icons are made automatically based on the type of data received, and validity checks are simply ignored. The real solution will require an automated interface to be developed for TAS.

Details of the design and implementation of the VS Demo have been documented in the IIPLESE Technical Information Report, *A "Vertical Slice" of the Intelligence Predictive Assessment System Concept (IPAS 2000)* [12]. The paper, "A Distributed System for Fully Automated Text Processing within the Intelligence Predictive Assessment System Framework", which was presented at Proceedings of the 4th Annual 1994 IEEE Mohawk Valley section Dual-Use Technologies and Applications Conference, May 23-24, 1994, describes an overview of the VS Demo.

### **2.3 Message Filtering with WIDE**

Experiments with the Warning Information Dissemination Experiment (WIDE) application had two major objectives. The first was to quantify the performance of the WIDE software as a message dissemination system with respect to both message throughput and accuracy. The second objective was to improve the performance of the system by 10% in throughput and by 15% in accuracy. Preliminary work was required to re-configure the delivered WIDE software, from a concept-building system for exploring a corpus of text to a message filtering system.

As originally delivered, the WIDE software was organized to provide for concept building, but not for message filtering. In the delivered software, users could create concepts by exploring a corpus of articles and selecting relevant or irrelevant ones. Concepts, which are essentially statistical profiles of the words in the messages marked relevant or irrelevant, with respect to the corpus, could be built and saved, but there were no mechanisms for applying the concepts to a stream of incoming messages. Under IIPLESE, the WIDE software was re-organized to provide a message filtering capability. Separate processes, based on the original WIDE code, were developed for zoning, part-of-speech tagging, profiling, and concept matching. The PRC Event Manager (eM) was used to link the processes together in a publish and subscribe framework. The re-configured WIDE software was used both for the VS Demo, described in the previous section, and the tests and improvements described below.

The approach to testing and improving the re-configured WIDE software involved several steps. First, application throughput was measured and improved. Then, using this more efficient implementation of WIDE, application accuracy, in terms of filtering recall and precision, was measured for a number of message data sets. After analysis of this data, modifications to improve the system accuracy were implemented and evaluated for a larger number of data sets. Finally, system throughput was measured again to account for changes that may have been introduced by improving the accuracy.

MUC-3 articles and a set of Time Magazine articles used for testing information retrieval algorithms were used to test the re-configured WIDE software. Though good results were obtained with these data sets, more testing of the WIDE software should be performed using the Text Retrieval Evaluation Conference (TREC) data, which provides a much larger corpus and set of evaluation queries.

The results obtained with improvements to the re-configured WIDE software reflected significant increases in throughput and accuracy. Using the entire MUC3 data for evaluation, the message throughput for the WIDE software was increased 150% to a capacity of 5,144 messages per hour. Most of this improvement was due to corrections to inefficient pieces of code.

Using the definition of accuracy as the product of recall and precision, measurements were taken using eight test queries from the MUC3 and Time Magazine data sets. Before any improvements, accuracy for the re-configured WIDE system varied from 0.02 to 0.4, with an average of 0.14. After modifications, accuracy values ranging from 0.25 to 0.91, with an average of 0.56, were obtained, representing an improvement of about 400%. The principal modification was allowing the relevance measure to take account of a threshold for the minimum number of occurrences of a word. (For example, with the threshold set to 3, words that occurred only once or twice in all of the messages marked relevant for a concept, would not be weighted in determining the relevance measure of an article for that concept).

The tests and improvements to the re-configured WIDE software are documented in the IIPLESE Test Report , *Warning Information Dissemination Experiment (WIDE) Testing, Evaluation, and Improvement* [11].

## 2.4 GIP Tests and Enhancements

In addition to being used for the VS Demo described above, the Generic Intelligence Processor (GIP) software was tested thoroughly as a standalone application. The GIP Application Builder (GAB) software was used to build a number of Message Processing Applications (MPAs) for 10 different message types. Overall, the GIP software gave satisfactory performance, though, as expected with a prototype system, several issues and bugs were raised by the testing; particularly with respect to the graphic user interface. These are documented in the IIPLESE Test Report, *Generic Intelligence Processor (GIP), Performance Testing and Assessment Report*, IIPLESE Technical Information Report (Test Report), M. Scully, Contract No.: F030602-91-C-0109, September 1994.

Several bug fixes and enhancements were made to the GIP software during the IIPLESE effort, in order to use it effectively for the VS Demo. An eM interface option was added to the Acquire module to allow users to build MPAs that acquire incoming message traffic via the eM. Several other minor bugs were fixed to make the software more robust: a problem with the way eM domain names were assigned, problems with using "system" commands to invoke other processes, and other minor problems.

For the VS Demo, a GIP MPA was developed and used to extract data from free text messages. This explored the issues of integrating GIP with other applications and exercised the GIP capabilities for NLU integration. The major integration issue for the VS demo was the requirement that the GIP MPA run on a single workstation. Though GIP uses the eM for some inter-process communication, it also uses shared memory to control start-up and shutdown of the seven major processes. This prevents the GIP processes from being distributed across several workstations, and reduces the ability to balance the overall processing load. The problem with NLU integration is that GIP really only provides a set of conventions for passing files to and from an NLU. The assumption is that the NLU application will return one or more completed templates for one or more types of events per message. No provisions are made for fusion or interpretation of the extracted data, which might best be done

in light of additional information provided by the NLU application. These issues should be addressed in the GIP follow-on effort, which, at the time of this report, has just started.

## **2.5 Speech Understanding Integration**

In the future, technology for both spoken and written language understanding will be an important part of any system for analysis of complex data. In light of the VS Demo that provides a prototype for application integration, an analysis was performed under IIPLESE to consider how spoken language understanding technology could be integrated within the IPAS 2000 framework.

The following subsections describe speech recognition technologies and the methods by which these technologies can be integrated into IPAS 2000. First, a technological overview of speech recognition is presented, including a brief survey of current capabilities and limitations. Then several selected speech recognition applications and environments are described, including one currently under development for Rome Lab. Lastly, three options are presented for incorporating today's state of the art speech recognition products into IPAS 2000 applications..

### **2.5.1 Speech Recognition Systems**

Speech recognition systems are distinguished by the capability of translating and processing electronic signals derived from the human voice. Among developers and users of speech recognition systems the term "speech recognition" is commonly taken to mean both speech recognition and speech understanding. Speech recognition is the process by which a computer maps an acoustic speech signal to text. Speech understanding is the process by which a computer maps an acoustic speech signal to some form of abstract meaning of the speech.

One of the main identifying attributes of a speech recognition system is the type of recognition performed with respect to the speaker. It is quite common for a system to be described as performing one of three types of recognition: speaker dependent, speaker adaptive, and speaker independent.

Speaker dependent systems are trained to operate for a single individual speaker. These systems offer the greatest degree of accuracy and are easiest to develop, but are not as flexible as other systems. Speaker adaptive systems incorporate a general model of speaker characteristics, and are designed to adapt recognition operations to new speakers. Speaker adaptive systems are generally more flexible than speaker dependent systems, but offer less accuracy. Speaker independent systems are designed to operate for any speaker of a particular type (e.g., English-speaking male). Speaker independent systems are difficult and expensive to develop, and are usually lacking in accuracy. Speaker independent systems are often more flexible than the other types.

A second identifying attribute of speech recognition systems involves the speech pattern used for recognition. There are two main speech patterns: continuous speech and isolated word. Speech recognition systems are often described according to speech pattern in exactly the same way as they are described according to recognition type. Thus one system might be described as providing "speaker independent continuous speech" recognition, while another system might be described as providing "speaker adaptive isolated word" recognition.

Isolated word systems operate on a single spoken word at a time. These systems require a discreet pause between each spoken word. The required length of the pause varies among individual systems. Isolated word recognition systems are significantly easier to develop than are continuous speech systems, but are unable to process normal spoken discourse.

Continuous speech systems operate on a series of "connected" words as occurs in normal human speech. Continuous speech recognition is significantly more difficult than isolated word recognition for several reasons. For example,

continuous speech processing requires that the speech recognition system correctly identify word boundaries. Continuous recognition can be adversely affected by varying rates of speech, and by other spoken phenomena such as coarticulation (coarticulation occurs when the pronunciation of a phoneme at the end of a word affects the pronunciation of a phoneme at the beginning of the next word).

The accuracy of existing speech recognition systems varies greatly with the style of speech and the mechanisms used to process that speech. Good speaker-dependent isolated word systems can achieve recognition rates of over 95% on large (>1000 word) vocabularies.

Speaker independent and continuous speech systems are generally less accurate than speaker dependent systems. For example, Voice Processing Corporation's VPro system performs continuous speaker independent recognition within an extremely limited domain (yes/no tokens and digits only). Recognition rates for a neutral male voice range around 90%, with a significant drop in recognition rates (below 70%) for accented speakers and children's voices, and an even more pronounced drop in the presence of background noise.

## **2.5.2 Speech Recognition Capabilities and Limitations.**

In speech recognition, as with all technologies, certain capabilities and limitations exist. A full listing of capabilities and limitations associated with the current state of the art is beyond the scope of this document, but selected capabilities and limitations which are pertinent to IRDS systems are briefly identified below.

The major capabilities are:

- Modern speech recognition systems are able to employ acoustic models that rank words according to similarities with a defined acoustic signal [17]. When the words are short (as in isolated word systems) and when the signals are well-defined (as in speaker dependent systems), a high degree of accuracy can be achieved.

- Modern systems which process spontaneous speech can get near real time performance on small tasks and small vocabularies (<1000 words) with reasonable accuracy. [18]

The major limitations are:

- Speech recognition systems generally cannot process unknown or out of-vocabulary words. [19]
- Speech recognition systems generally exhibit poor performance when background noise is present. Also, these systems tend not to degrade gracefully. [18]
- Speech recognition applications are not easily ported to new and unfamiliar domains.

### **2.5.3 Speech Recognition Products and Environments.**

This section provides a brief description of several selected speech recognition products and environments. In this context, the term "product" refers to speech recognition software (and, possibly, hardware) which is meant to perform a specific task. The term "integrated technology" refers to certain available speech recognition tool suites which serve as development laboratories for spoken language systems. Speech recognition environments (sometimes called "Speech Laboratory Environments") generally provide functionality for most aspects of speech processing, including application development. Speech laboratory environments are commonly developed for high-powered workstations and midrange computers.

Some of the applications surveyed are produced by commercial organizations and may be quite expensive. Other applications are produced by universities and/or other research organizations and are free. This section is intended to give the reader a brief overview of the types of products and technologies currently available. It is not meant to provide detailed product evaluations.

#### 2.5.3.1 IN3 Voice Command.

IN3 Voice Command is a commercial product which provides a continuous speech recognition facility for the SunOS or Solaris operating systems. The recognition system is a secure operating system facility capable of working with various interfaces, microphones, and devices. The user interface provides a means to quickly create commands on the fly for replacing long strings and complex operations with voice macros.

#### 2.5.3.2 HM2007 Speech Recognition Chip.

HM2007 is an inexpensive 48-pin single chip CMOS voice recognition LSI circuit with on-chip analog front end, voice analysis, recognition process and system control functions. A 40 word isolated-word voice recognition system can be composed of an external microphone, keyboard, SRAM and a few other components. When combined with a microprocessor, an intelligent recognition system can be built.

#### 2.5.3.3 Entropic HTK Toolkit.

HTK is a software toolkit for building continuous density Hidden Markov Model based speech recognizers. It consists of a number of library modules and a number of tools. Functions include speech analysis, training tools, recognition tools, results analysis, and an interactive tool for speech labeling.

#### 2.5.3.4 DragonVoice Tools.

DragonVoice Tools is a commercial programmer's toolkit for developing speech-aware DOS or Windows applications for 80X86 microcomputers. Recognizes continuously spoken digits and discretely spoken words or phrases. Up to 1,000 words can be active at one time. The software uses words from a 110,000 word dictionary, and allows developers to produce new word models.

#### 2.5.3.5 BBN's HARK™/Delphi Systems, and the SPLINT Domain.

BBN Systems & Technologies (under contract to Rome Laboratory) is exploring an interactive spoken language interface to a SYBASE database. The HARK™ system provides real-time, speaker independent speech recognition, and the Delphi system provides a language understanding functionality. [4]

These two systems are used as speech understanding components of a project currently referred to as SPLINT (Speech and Language Integration). Strictly speaking, SPLINT refers to a very specific information domain. The SPLINT project aims to provide a spoken language query interface to a database containing specific data on Air Force units and equipment at Griffiss AFB, N.Y. and Langley Field, VA. [5]

SPLINT is expected to be delivered to Rome Laboratory in October 1994.

#### 1.3.2.1 Entropic Signal Processing System (ESPS)

ESPS is a commercial environment distributed by Entropic Research Laboratory of Washington, D.C. ESPS incorporates a variety of speech analysis tools, a C language programming library, Hidden Markov Modeling (HMM) toolkits, and a graphical user interface. ESPS allows the analyst to process pitch traces, speech waveforms and spectrograms, and supports signal labeling in relation to speech databases.

#### 2.5.3.2.2 OGI Speech Tools.

OGI Speech Tools is a free environment developed and distributed by the Center for Spoken Language Understanding (CSLU) at the Oregon Graduate Institute. OGI Speech Tools are intended for use in UNIX environments and include time-synchronous display tools, C library routines for speech data manipulation, a vector quantizer, conversion utilities, database utilities, a neural network training package, and a graphical user interface.

#### **2.5.4 Integrating Speech Recognition Into IPAS 2000 Applications**

It is currently possible and feasible to integrate speech recognition technologies into existing IPAS 2000 applications. This section presents three implementation options and describes some of the benefits and drawbacks associated with each option. The Vertical Slice Demonstration (VS Demo) is used as an example system to illustrate some of the points presented with each option. It should be noted that the options presented below are not necessarily specific to the VS Demo, but may also apply to a number of applications, whether integrated within the VS Demo framework or not.

The first, and simplest, option is to integrate an existing product into an existing system. The specific product to be integrated is BBN's SPLINT, which was described previously. Integration of SPLINT "as is" into the VS Demo should pose no technical difficulty for the integrator. As proposed by BBN; SPLINT functions as a standalone application which serves as an interface to an existing database. As a standalone, SPLINT can simply be grafted into the VS Demo's workspace. If the delivered SPLINT user interface is text-based it can be run within its' own window in the VS Demo environment via a remote shell. If the delivered SPLINT user interface is X Windows-based, the integrator can perform minor modifications to the VS Demo initialization routines to achieve SPLINT startup within the common environment.

Integration the SPLINT functionality presents more of a challenge. The SPLINT domain is very specific as regards aircraft, ordnance, equipment and locations. Again using the VS Demo as an example, it would appear at first glance that the data processed by the VS Demo is of a different type than the data residing in the SPLINT domain. A SPLINT product would appear within the context of the VS Demo as a proof-of-concept utility that has no impact on the rest of the system's function. The user might perform VS tasks, or might query a database via SPLINT, but not both.

To achieve some concordance of functionality, the VS Demo would need to be slightly enhanced to process messages within the SPLINT domain, and to update SPLINT-readable databases. By doing this the technologies would be

more closely coupled; SPLINT could now be used to query a database that other VS Demo utilities have updated. The enhancements needed for the VS Demo to process SPLINT-domain data would require modification of the VS MPA's "format" and "forward" routines, a new or revised Message Filter, and new or revised database interfaces processes.

The second option is to modify an existing product and then integrate it into an existing system. As with the previous option, the specific product to be integrated is BBN's SPLINT. As was noted above, the SPLINT domain does not correspond well with the data types processed by the VS Demo. However, the SPLINT database mechanisms do appear to be well-matched to the database mechanisms used in the VS Demo. An obvious course, then, is to apply the SPLINT speech recognition technology to a different recognition domain. This is clearly a tractable problem. In [21], the authors describe just such an effort in creating the SPLINT domain, based upon the speech recognition technology used in the ATIS system.

A domain appropriate for the VS Demo might incorporate events, times and locations rather than the aircraft, ordnance and equipment used in the SPLINT domain. For example, the following spoken queries are valid within the SPLINT domain:

- "What is the range of the AGM-65C Maverick missile?"
- "How many Air Force bases are there in the US military area?"
- "Show me a map of Griffiss."

Queries of a similar logical type can be constructed for an "event domain" such as might be used for the VS Demo:

- "How many terrorist events occurred in Cuba over the last month?"
- "Sort surface ship tracks by time of day."
- "Show me the text of any news report related to aircraft hijackings."

Creation of a new domain using SPLINT technology would require more time and effort than a simple SPLINT integration, and may require the cooperation of the original SPLINT developer.

The third option is to achieve a harmonious fusion of applications and speech recognition technologies, independent of any one specific speech recognition product. This option is more complex than the previous two, but may provide useful near-term and long-term benefits.

Using the VS Demo as an example, we see that existing speech recognition technologies can be integrated and used in at least three ways - with spoken command interfaces, spoken application interfaces, and spoken database interfaces.

A spoken command interface is one that translates speech to command imperatives at the system level. System-level spoken commands would tend to be brief, and would cover a range of functions at the operating system level. For example:

"Start message processing"  
"Use terrorism configuration"  
"Close text editor"

Spoken command interfaces can be implemented rather cheaply and quickly. There are a great many existing COTS products that perform exactly this function, and it is also comparatively easy to develop these interfaces in-house by using an integrated speech processing tool. The majority of COTS products that provide spoken command recognition are speaker-adaptive or speaker-dependent isolated word systems, and are therefore reasonably accurate.

A spoken application interface is one that translates speech to command imperatives at the application level. This can be illustrated with examples using the various applications from the VS Demo:

Message Processing  
Application:

"Start factbase editor"  
"Select all text"  
"Cut"  
"Paste"  
"Iconify"

Manual NLU:

"Show the message body"  
"Copy the event field to output template one"

Scenario Driver:

"Load COMSPOT messages"  
"Send a message every thirty seconds"

For any application to have a useful spoken application interface, it would be necessary to modify existing systems' source code. Therefore, this kind of interface might be most useful with GOTS and in-house software rather than commercial software. Speech recognition development tools would facilitate development of the interface.

A spoken database interface would provide the user with a query facility for VS Demo databases, and would complement the command and application interfaces. The SPLINT application provides an interface to a sample data base, but, as delivered, cannot be adopted to handle other databases. Speech recognition development tools and environments will be needed to adopt or develop a spoken language interface for a specific database, and to maintain the interface as the data base or types of queries change.

## 2.6 ART/AMUS Integration and Testing

Under IIPLESE the ART Message Understanding System (AMUS) was integrated with GIP, using the experience gained from the earlier integration of PLUM with GIP. The GIP MPA for the VS Demo, the VS MPA, that handles message processing for PLUM was augmented to handle message processing for AMUS. The utility of the data extracted from text via AMUS, however, requires further analysis.

Modification of the VS MPA required defining an additional message type, SPOT\_AMUS. Though SPOT messages were already handled by the VS MPA for PLUM, AMUS required only the text portions (English sentences) of the message as input, whereas PLUM required the full message content. As a result different FParse message maps were required, and this necessitated using different message types. Solutions to this problem would be to either allow multiple message maps per message type in FParse, or modify or require NLU applications to accept the same breakout of the message as input. In addition PLUM and AMUS provide entirely different templates or vectors of data extracted from SPOT messages. For example, PLUM provides flight path data as a list of coordinates for the FLIGHT\_PATH field in a single output template, whereas AMUS provides each segment (between two coordinates) of a flight as a separate vector of data. As a result, the Select, Translate, Format, and Forward modules of GIP must treat the extracted data separately - as if there were separate GIP clients to handle it.

The interface between GIP and AMUS uses the eM to provide more sophisticated control over start-up and exchange of data between the two processes. The GIP - PLUM interface used files to transfer data between PLUM and the GIP UParse module. Remote copy commands (rcp) were used to transfer PLUM output on one workstation to files loaded by UParse on another workstation. In addition to being slower, this interface lacks flexibility, since it must take account of file access permissions and specific network hosts. The GIP - PLUM interface also relied on the internal queuing mechanisms of the eM to handle data to be passed to PLUM. Since the PLUM processes run much slower than the GIP processes, and cannot process data as fast as GIP can

provide it, this is not a robust approach. Client processes (of the eM) should manage their own queues of data, to provide graceful degradation, or intelligent handling of different processing rates.

The GIP - AMUS interface uses an eM interface process to manage a queue of data being exchanged between the processes. AMUS was modified to subscribe to eM events and interpret them as commands or data to be analyzed. An eM process was written that handles input and output to/from the UParse module, and provides the interface to AMUS. To handle the difference in processing speed, since, like PLUM, AMUS is much slower than the GIP processes that are feeding it data, the interface process maintains a queue of data to be processed by AMUS. When AMUS completes processing of a piece of text, the interface process automatically provides it with the next item on the queue. Whenever the UParse process provides a notification of data to be processed by AMUS, the interface process adds it to the queue. In addition, the interface process handles commands to initialize and shutdown AMUS.

Output from AMUS essentially takes the form of one or more vectors from each sentence in the message. Usually each SPOT message sentence results in one or more pairs of vectors. Each pair consists of a Generic vector and a Flight vector. The Generic vector describes the type of event taking place, and the Flight vector describes a line segment of the flight path. For messages with simple, short sentences about a single flight, the vectors produced by AMUS describe events reasonably well. However, small changes in the wording of sentences were observed to produce drastic changes in output vectors. In addition, compound sentences were not handled well. Though more thorough testing is needed, these problems made it difficult to use anything but very simple data with AMUS.

### 3. Additional Accomplishments

In addition to the tasks described in section 2, many other smaller tasks or experiments were performed to investigate or provide the basis for investigation of different aspects of the IPAS 2000 architecture. These tasks can be summarized under the following categories

Configuration management to support in-house IRD work. For example, installation, maintenance, evaluation and demonstration of OTS and contractor software in the ISF.

Investigations into use of IDHS databases to support Program 6 applications. For example, interfaces between DAWS and TAS.

WSS integration. For example, TAS and GIP were integrated within WSS. Assistance was provided in integrating PSIDS.

In addition to several technical reports, several presentations were made, and one paper published, describing work under IIPLESE. These are:

"Lessons Learned during IIPLESE Testing and Integration Experiments", S. Barth, presentation at the 1993 RL/IRD Technical Interchange Meeting, February, 1993.

"A Distributed System for Fully Automated Text Processing within the Intelligence Predictive Assessment System Framework", S. Barth, J. Penatzer, and J. Pirog, presentation at the 1994 RL/IRD Technical Interchange Meeting, March, 1994.

"A Distributed System for Fully Automated Text Processing within the Intelligence Predictive Assessment System Framework", S. Barth, J. Penatzer, and J. Pirog, Proceedings of the 4th Annual 1994 IEEE Mohawk Valley section Dual-Use Technologies and Applications Conference, May 23-24, 1994, pp 270 - 276.

"A Distributed System for Fully Automated Text Processing within the Intelligence Predictive Assessment System Framework", S. Barth, J. Penatzer, and J. Pirog, presentation at the Symposium on Advanced Information Processing and Analysis (AIPA94), 22- 24 March, 1994.

## 4. What's Next

---

The tests, experiments, and development of the IPAS 2000 architecture under the IIPLESE effort have provided a solid basis for further in-house and contractor work for RL Technology branch. Some of the major ideas for further experiments and development are listed below.

*Provide configuration compatibility between ISF workstations and the AF Client-Server Environment. CSE.* This would aid fieldability of Program 6 applications, as CSE becomes the standard workstation environment.

*Provide On-line Configuration Data for the ISF and IIPF Program 6 areas.* Though small these LANs change frequently. On-line configuration data could be made available to contractors over the InterNet (via Mosaic, for example) to facilitate delivery of software to RL.

*Apply the VS Demo to classified data.* Incorporate IDHS databases. Develop or customize an NLU application for classified messages.

*Compare message filtering techniques.* The modified WIDE software achieved good results on a limited data set; however further testing and comparison with TREC results is necessary.

*Use QSP as the mechanism for interfacing with IDHS databases for the VS Demo (in the ISF and IIPF).* This should make the task of application integration easier by providing a common DBMS interface.

*Integrate the VS Demo (in the ISF) with an Object-Oriented Database Management System (OODBMS).* Develop interfaces from GIP to the OODBMS.

*Address the issues of data fusion for partial and incomplete results from NLU data extraction.* Can multiple NLU applications be used to improve results? Can data fusion be performed as an independent process operating on NLU output?

*Provide a seamlessly integrated user interface for IPAS 2000.* Include access to multimedia, access to the InterNet and WWW, video conferencing, and applications for processing and analyzing data. The prototype interface should have the characteristics of the multimedia user interfaces that are expected to emerge in the next few years.

These, and many other ideas from work initiated under IIPLESE, will enable full realization of the IPAS 2000 concept, and greatly enhance the productivity and performance of Intelligence Analysts.

## 5. References

---

1. *Intelligence Predictive Assessment System (IPAS 2000) Research, Development, and Transition*, Version 1.2, Draft, John Salerno, John Pirog, Capt. Bill Shobert, Lt. Eric Jumper, Capt. Randy Douglass, IRD Technology Branch, 15 June 1992.
2. *Advanced Reasoning Theory, Final Technical Report*, RL-TR-93-2, Harris Corporation, J. Reed, N. Heyda, Contract No.: F30602-90-C-0020, March 1993. \*
3. *Generic Intelligence Processor (GIP), Final Technical Report*, RL-TR-93-181 Sterling ITD, W. Reid, D. Gray, and A. Lazzara, Contract No.: F30602-91-C-0097, October 1993. \*
4. *Temporal Analysis System (TAS) User's Manual*, GTE, 1992.
5. *Final Report, Warning Information Dissemination Experiment, (WIDE)* Systems Research and Applications, TR-92-618-0157, December 1992.
6. "The Event Manager Toolkit", Ladwig, M. PRC's Technology Transfer, Vol. 5, Number 6.
7. *Event Manager Architecture for Distributed Processing*, M. Ladwig, Technical Report TR-RD-93-1, 1993, PRC, McLean, VA.
8. "BBN PLUM: MUC-3 Test Results and Analysis", - R. Weischedel, D. Ayuso, S. Boisen, R. Ingria, and J. Palmucci. Proceedings of the Third Message Understanding Conference (MUC-3), May 1991. Morgan Kaufmann Publishers Inc., San Mateo, CA
9. *S/W Design Document for the DAWS Database Support (DBS) CSCI*, Working Paper 0131, CDRL D026, D032, 30 Sept. 91.
10. *Generic Intelligence Processor (GIP), Performance Testing and Assessment Report*, IIPLESE Technical Information Report (Test Report), M. Scully, Contract No.: F030602-91-C-0109, September 1994.
11. *Warning Information Dissemination Experiment (WIDE) Testing, Evaluation, and Improvement*, IIPLESE Technical Information Report (Test Report), J. Penatzer, Contract No.: F030602-91-C-0109, September 1994.
12. *A "Vertical Slice" of the Intelligence Predictive Assessment System Concept (IPAS 2000)*, IIPLESE Technical Information Report (Test Report), S. Barth, J. Penatzer, Contract No.: F030602-91-C-0109, September 1994.

\*Although this report references limited documents listed above, no limited information has been extracted. Distribution authorized to U.S. Government Agencies and their contractors; critical technology.

13. "A Distributed System for Fully Automated Text Processing within the Intelligence Predictive Assessment System Framework", S. Barth, J. Penatzer, and J. Pirog, Proceedings of the 4th Annual 1994 IEEE Mohawk Valley section Dual-Use Technologies and Applications Conference, May 23-24, 1994, pp 270 - 276.
14. *IIPF Expert System Experiments (IIPLESE) Technical Information Report (Test Plan), Inventory and Configuration Management to Support Unclassified IIPLESE Testing*, S. Barth, M. Scully, J. Penatzer, Contract # F30602-91-C-0109, CLIN 0002, Data Item A002, 30 September 1994.
15. *IIPF Expert System Experiments (IIPLESE) Technical Information Report (Test Plan), Inventory and Configuration Management to Support Classified IIPLESE Testing*, S. Barth, M. Scully, J. Penatzer, Contract # F30602-91-C-0109, CLIN 0002, Data Item A002, 30 September 1994.
16. *Technical Information Report: Interface Control Document (Final)*, B. Buteau, Cooperative Knowledge Base Architecture (CKBA), Contract No F30602-90-C-0127, CLIN 0002, ELIN A0002, January 19, 1993.
17. J. Ueberla, "State Language Models for Speech Recognition", Report CMPT TR 93-03, Simon Fraser University, Burnaby, B.C., Apr. 7, 1993, pp. Y-Z.
18. R.A. Cole and L. Hirschman et al., Workshop on Spoken Language Understanding, Oregon Graduate Institute Technical Report No. CS/E 92-014, Sep. 1, 1992, pp. Y-Z.
19. S.R. Young, *Learning New Words from Spontaneous Speech: A Project Summary*, Carnegie Mellon University Technical Report No. CMU-CS-93-223, Jul. 29, 1993, pp. Y-Z.
20. D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund, M. Przybcki, "1993 Benchmark Tests for the ARPA Spoken Language Program", *Proceedings of the Human Language Technology Workshop (Weinstein, ed.)*, Morgan Kaufmann, 1994.
21. M. Bates, "Beginning to Port a Spoken Language Database Interface", *Proceedings of the 1994 IEEE Dual-Use Technologies and Applications Conference*, May 1994, pp. 278-285.

***MISSION***  
***OF***  
***ROME LABORATORY***

**Mission.** The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

- a. Conducts vigorous research, development and test programs in all applicable technologies;
- b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;
- c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;
- d. Promotes transfer of technology to the private sector;
- e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.